Problem B. Buggy Combination Lock

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

You're a full-time bomb defuser. Your duties include keeping talking and not letting anyone explode.

This time, you're stumbled upon the last obstacle before defusing the bomb — a combination lock. The lock contains n rotating discs, with each of the discs containing all integers between 0 and $m-1$, inclusive, in increasing order. One forward rotation of a disc causes the number on it to increase by one (except when the number on the disc is $m-1$, it changes to 0). Similarly, one backward rotation of a disc causes the number on it to decrease by one (except when the number on the disc is 0, it changes to $m-1$).

You see the initial state of the lock, and you perfectly know the code combination which opens the lock. Unfortunately, the lock is buggy. Whenever you rotate the i-th disc forward or backward, the $i+1$-th disc gets rotated in the same direction as well. Similarly, whenever you rotate the n-th disc, the first disc gets rotated in the same direction too.

On one hand, this might help you open the lock sooner; on the other hand, this might prevent you from opening the lock at all; who knows? Well, of course you do. Find the minimum number of disc rotations you need to perform to open the lock, or determine that it's impossible (and someone is about to explode).

Input

The first line of the input contains two integers n and m ($2 \le n \le 2 \cdot 10^5$; $2 \le m \le 10^9$) — the number of discs in the lock and the range of numbers on the discs, respectively. The second line contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i < m$) — the initial numbers on the first, second, ..., n-th disc. The third line contains n integers $b_1, b_2, \ldots, b_n$ ($0 \le b_i < m$) — the target numbers on the first, second, ..., n-th disc.

Output

If it's impossible to open the lock, output -1. Otherwise, output a single integer — the minimum number of disc rotations you need to perform to open the lock.

Examples

standard input	standard output
6 3 0 1 0 1 0 1 1 0 1 0 1 0	4
3 7 4 2 1 1 2 5	3
2 10 7 7 3 5	-1

Note

In the first example test case, one possible solution is to rotate the first and the second discs forward, both once, and the fourth and the fifth discs backward, both once.

In the second example test case, the fastest way to open the lock is to rotate the third disc backward three times.

In the third example test case, whichever disc you rotate, the numbers on the discs will always remain equal.